ADT PLM

Programmer's Learning Machine

Matthieu Nicolas

IJD Seminar, 2016-02-02

Outline

Outline

Presentation of PLM

Purposes

- Application to learn programming.

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.

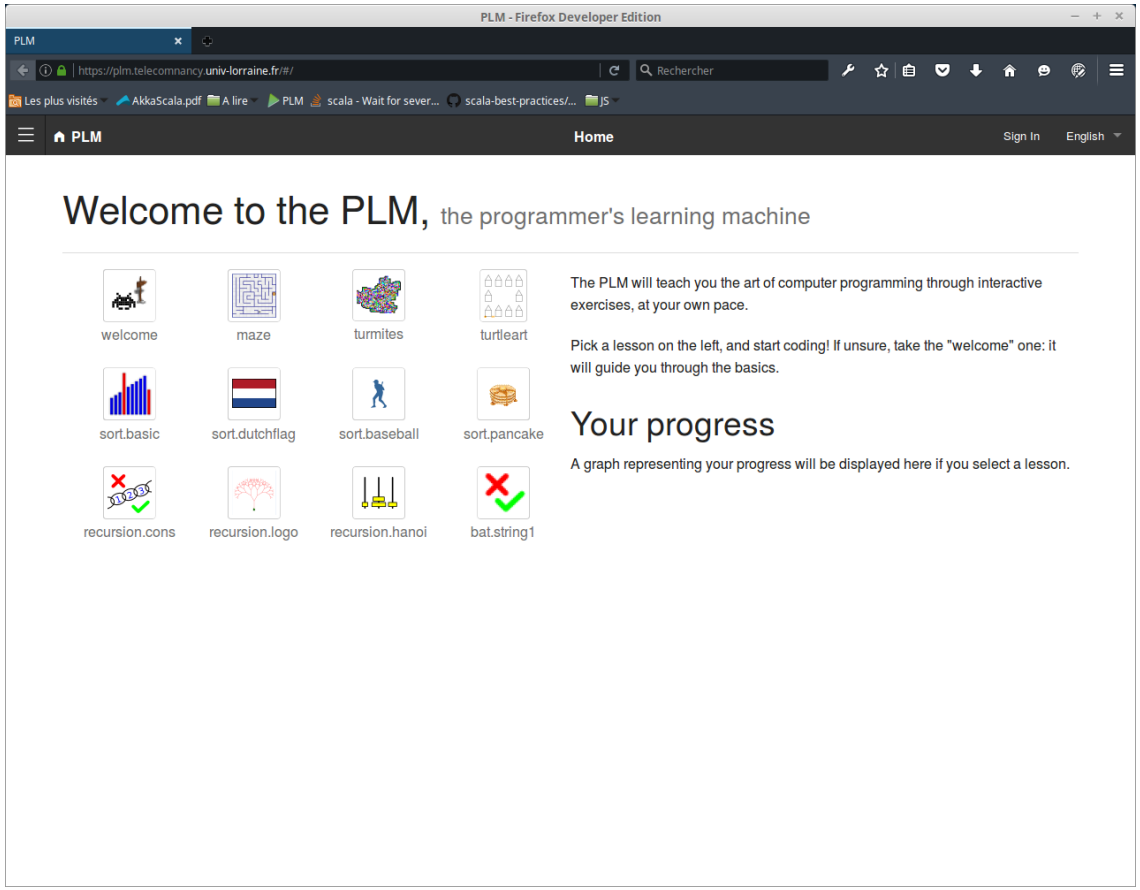
Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.
- Used at TELECOM Nancy since 2008.

Presentation of PLM

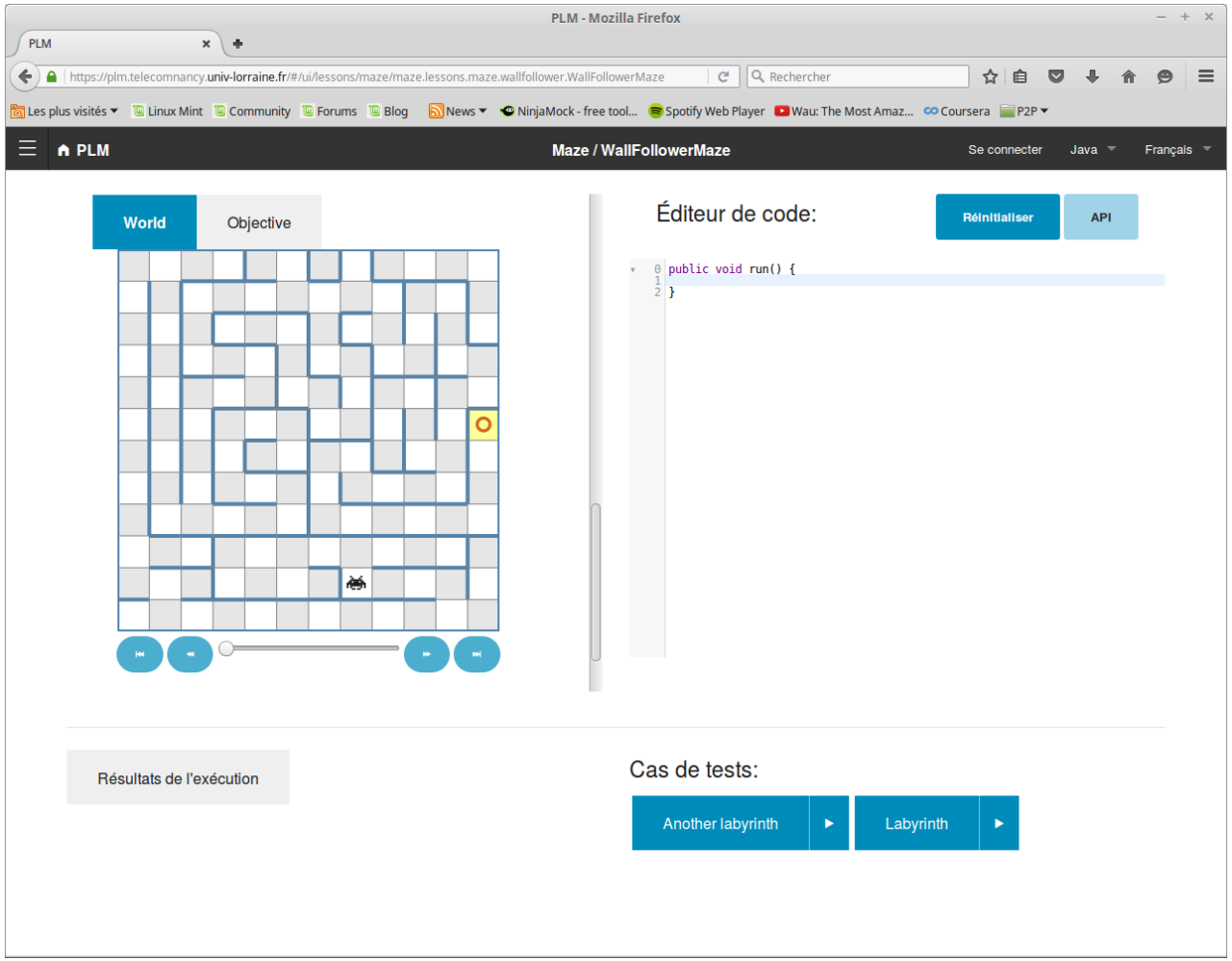
Quick demo



- Available at `https://plm.telecomnancy.univ-lorraine.fr`

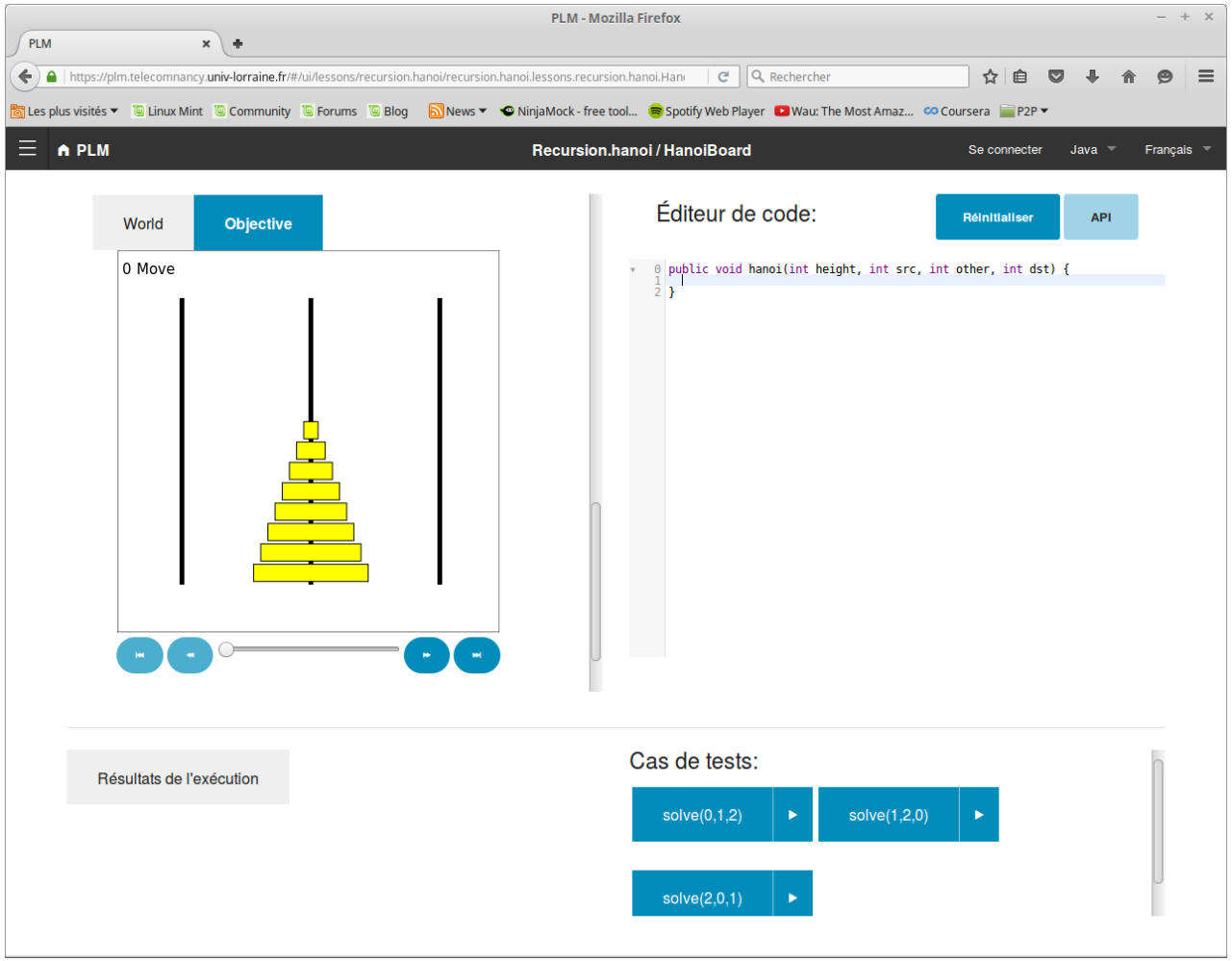
Presentation of PLM

12 lessons, 200 exercises



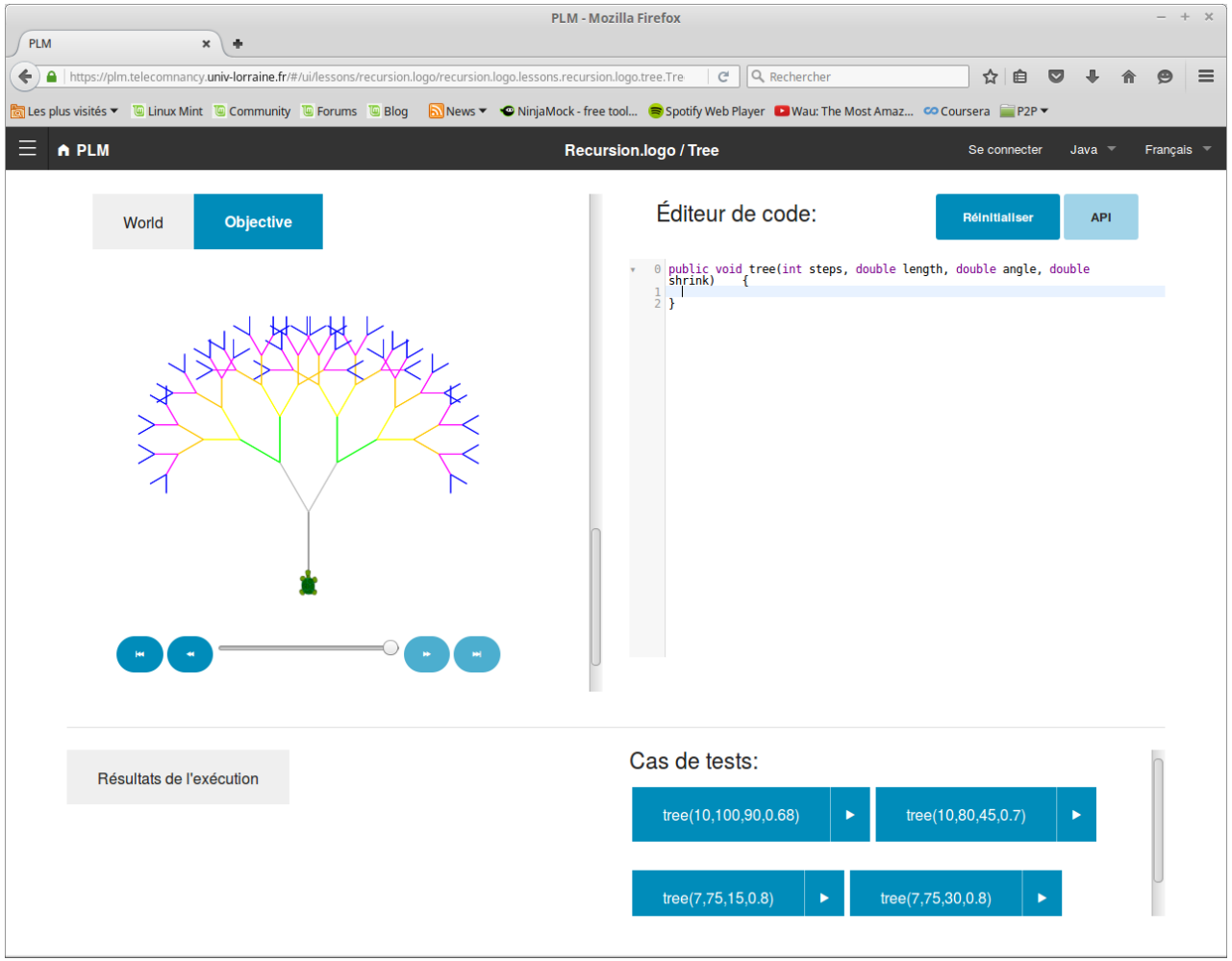
Presentation of PLM

12 lessons, 200 exercises



Presentation of PLM

12 lessons, 200 exercises

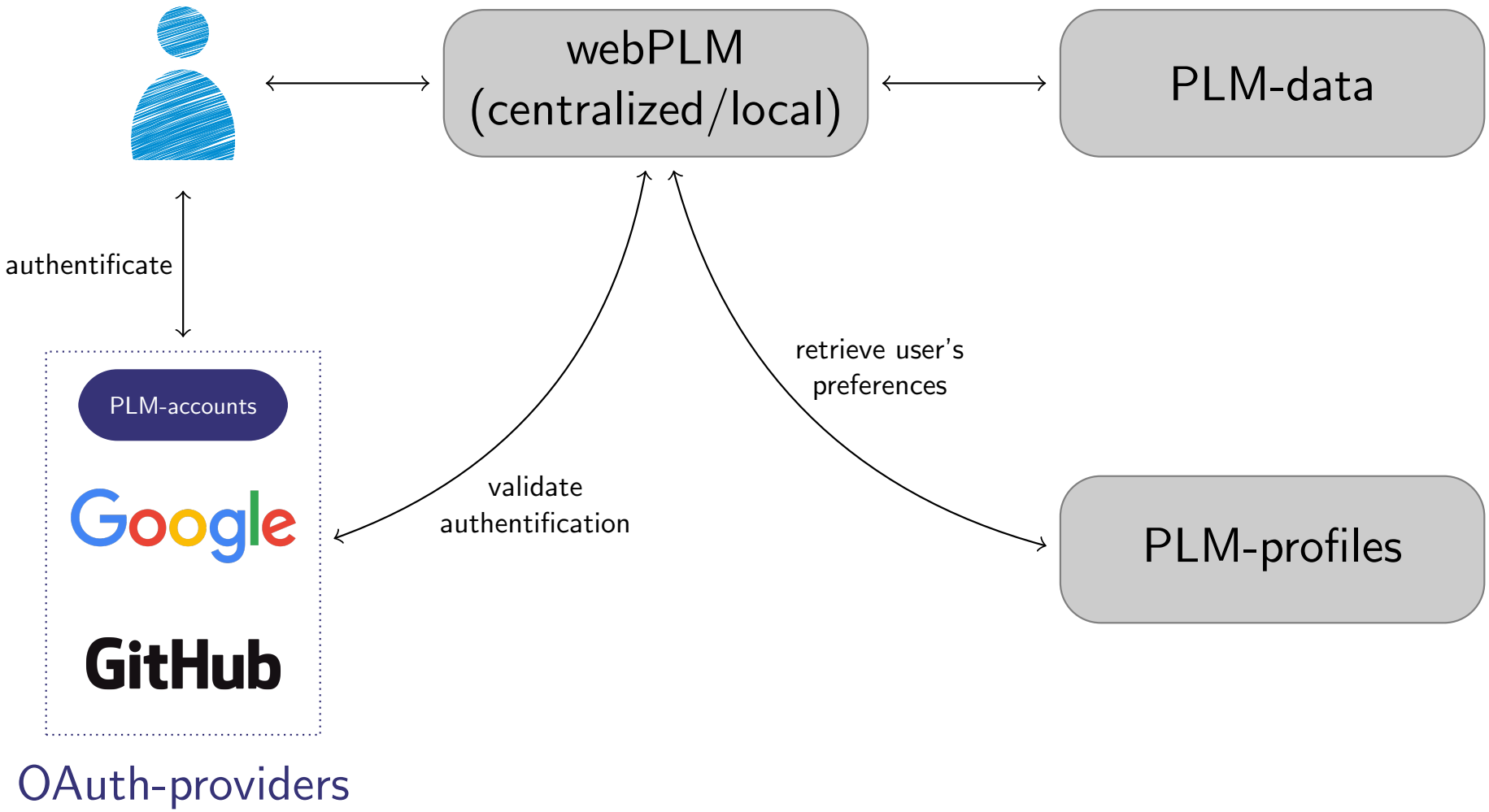


- Available languages:
  - English
  - French
  - Brazilian Portuguese

- Supported programming languages:

Presentation of PLM

Application's architecture



- Keep track of the users' progress...

- Keep track of the users' progress...
- ... using a git repository

Presentation of PLM

How does it work?

- Store users' code versions

Presentation of PLM

How does it work?

- Store users' code versions
- Store users' actions as commit messages

- Working in anonymous branch
- Branch pushed to a **GitHub** repo

Outline

To a web app

Evolution of the project

- Formerly a fat client
  - Written in Java

To a web app

Evolution of the project

- Formerly a fat client
  - Written in Java
- Switch to a web application
  - Server implemented in Scala using *PlayFramework*
  - User interface written in Javascript using *AngularJS* and *Foundation*

To a web app

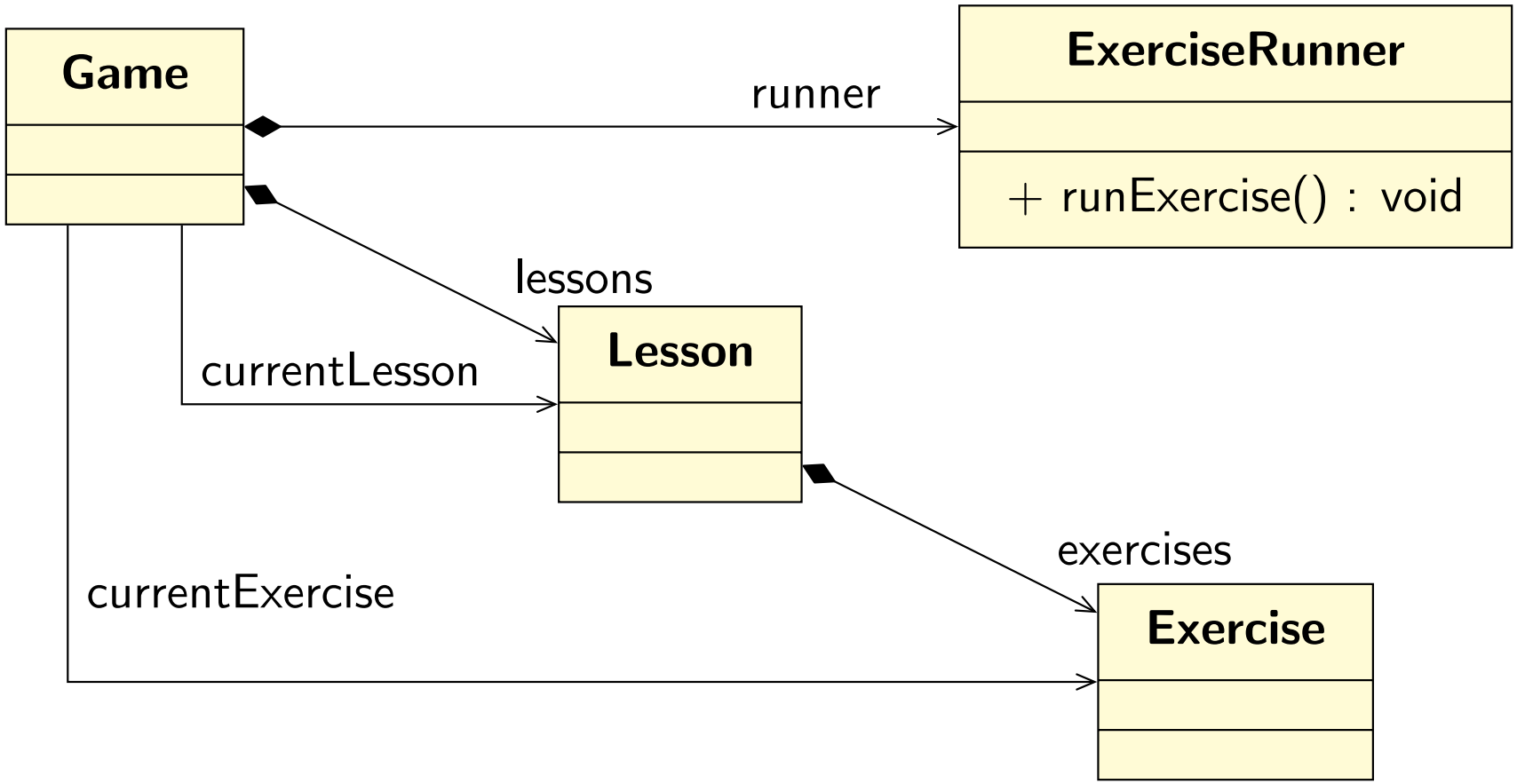
Motivations

- Want to switch to SaaS[1]
  - Easy to use
  - Easy to update
  - Easy to track usage data
- More user-friendly
- Aim to setup SPOC[2] and MOOC[3]

- But don't have the time and means for a reboot

[1] Software as a Service
[2] Small Private Online Course
[3] Massive Open Online Course

- Implemented a headless version of PLM: *PLM-engine*
  - Provide all PLM's content and methods
  - But without a user interface



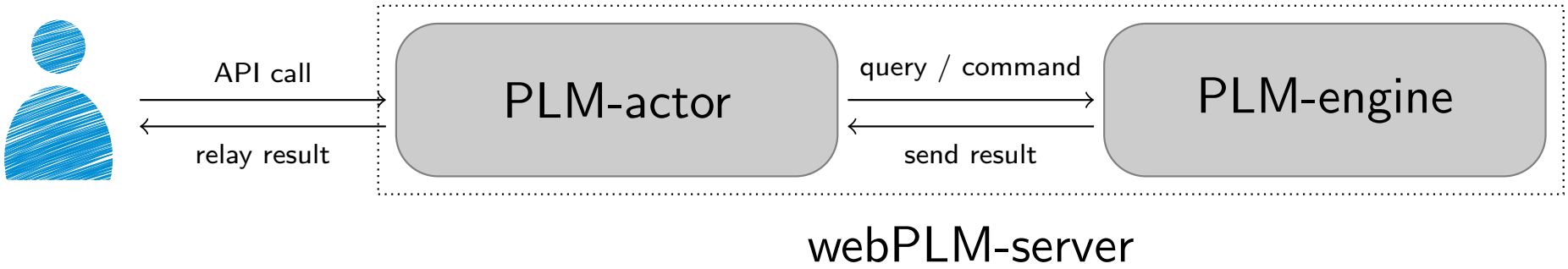
To a web app

Implementing the server

- Designed an API over PLM-engine
- Only need to implement a controller
  - Verify calls received from the client
  - Query or command PLM-engine according to the call
  - Send back result or acknowledgement to the client

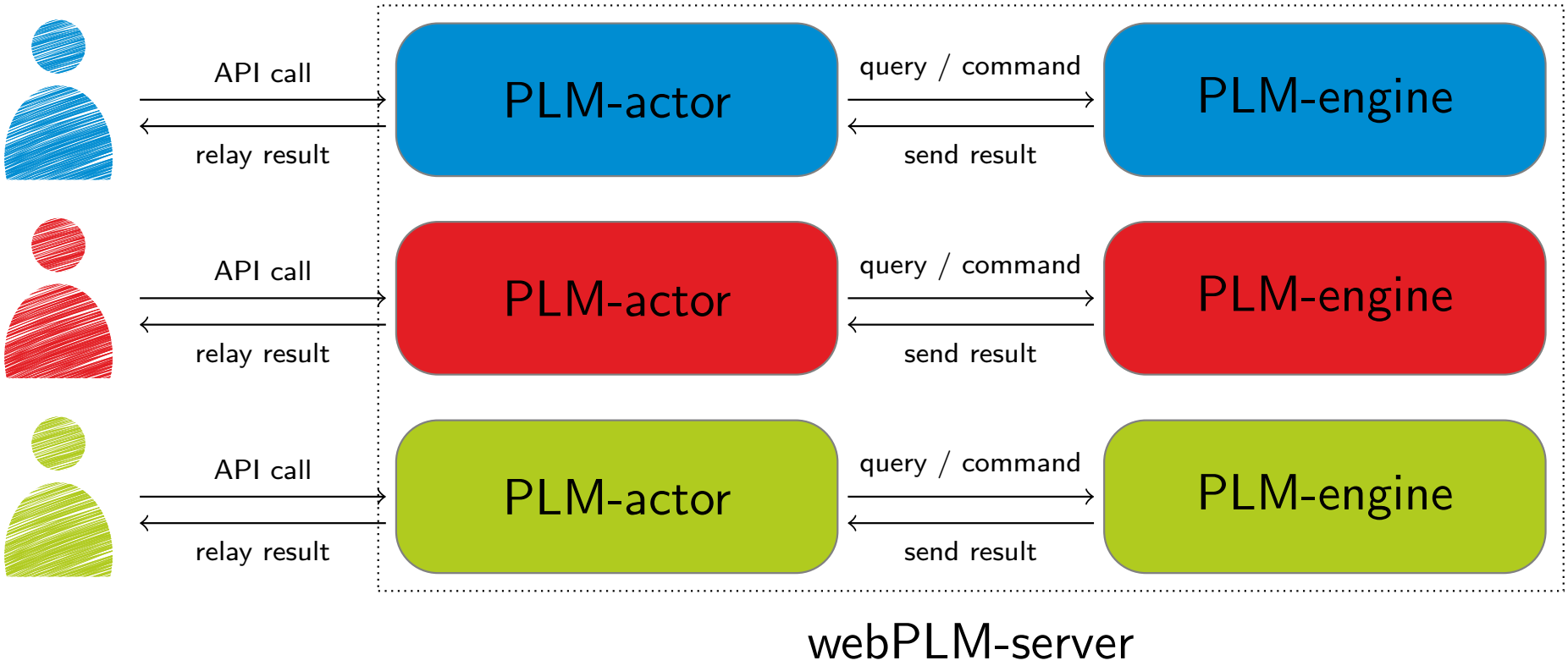
To a web app

Interactions between components



To a web app

Multi-user scenario

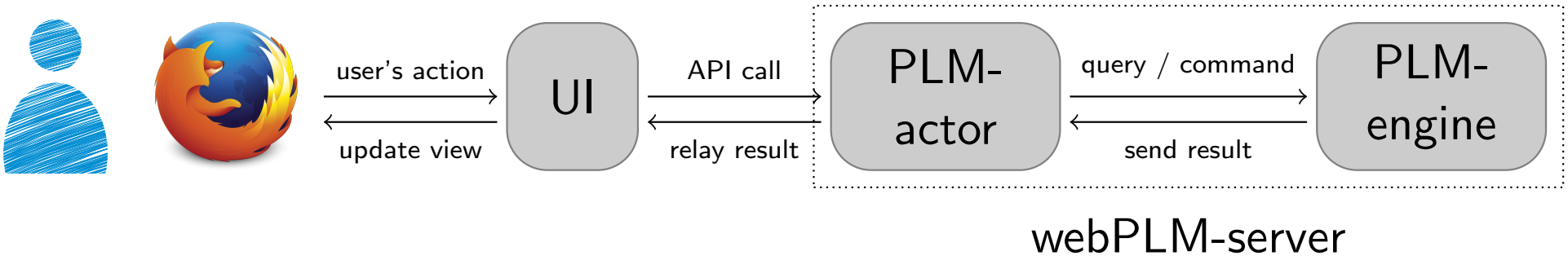


To a web app

Results

- Build quickly a web server from the fat client...
- ... but we also need a user interface

To a web app


user's action
update view
UI
API call
relay result
PLM-
actor
query / command
send result
PLM-
engine
webPLM-server

To a web app

- Have to translate user's actions into API calls
- Have to re-implement PLM-engine's data models

Outline

Assessment of user's code

Limits

- Run on the same machine, same JVM

Assessment of user's code

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops

Assessment of user's code

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation

Assessment of user's code

Limits

- Run on the same machine, same JVM

- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation
- And from *System.exit(whatever)*?

Assessment of user's code

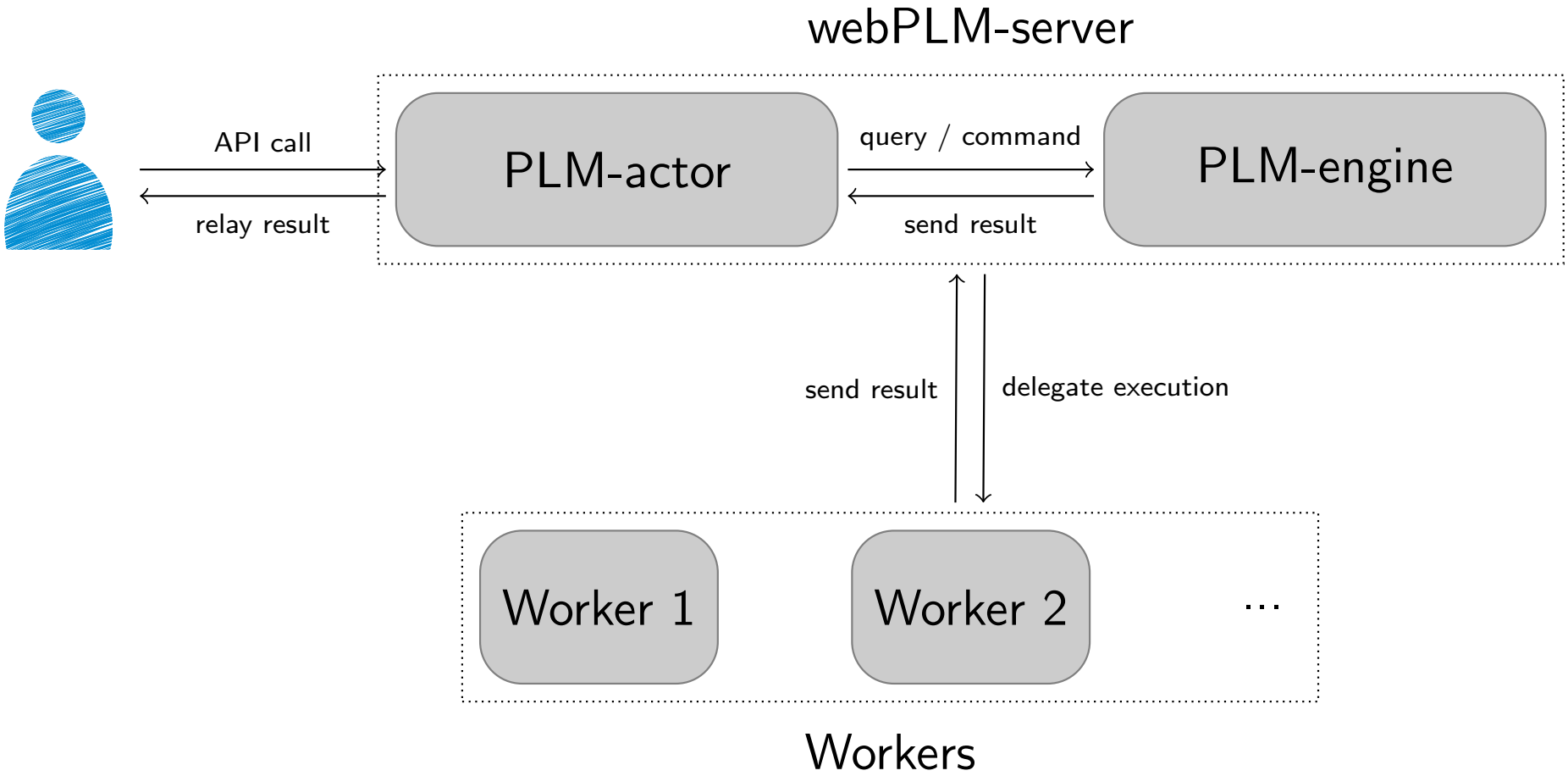
Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation
- And from *System.exit(whatever)*?
- Scalability issues

Assessment of user's code

Chosen solution

- Delegate execution to workers



Assessment of user's code

The judges

- Called *Judges* in the litterature
- Use PLM-engine as well

- Workflow:
  - Retrieve an execution request
  - Parse the request to extract parameters
  - Configure PLM-engine according to them
  - Run the user's code
  - Send back result to webPLM

Assessment of user's code

Message queues

- Message-driven architecture
- Loosely coupled system
- Asynchronous/Synchronous
- Help to implement:
  - Producer/Consumer pattern
  - Request/Response pattern
- Different reliability patterns of the message processing:
  - Only one worker
  - At least one worker
  - All workers
- Easy to scale

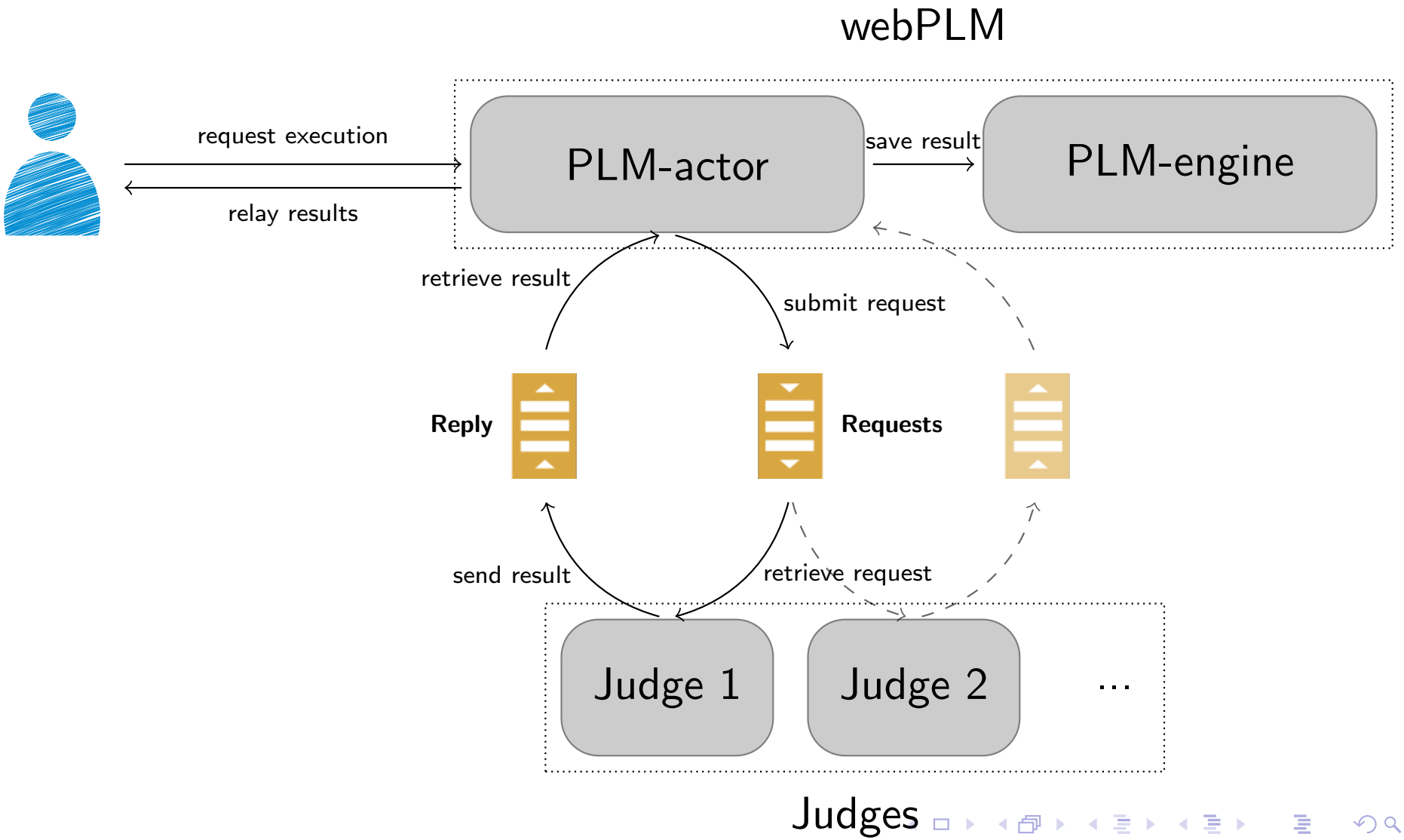
Assessment of user's code

Message queues

- Message-driven architecture
- Loosely coupled system
- Asynchronous/Synchronous
- Help to implement:
  - Producer/Consumer pattern
  - Request/Response pattern
- Different reliability patterns of the message processing:
  - Only one worker
  - At least one worker
  - All workers
- Easy to scale

Assessment of user's code

Architecture with judges



Assessment of user's code

Pros and cons

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements

Assessment of user's code

Pros and cons

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements
- Cons:
  - Make sure to use the right version of PLM-engine
  - Need to deploy them easily
  - Should restart them after each execution
  - Have to restrict their resources usage

Assessment of user's code

Docker

- Lightweight virtualization tool
- Build image of your application
- Run containers based on images

Assessment of user's code

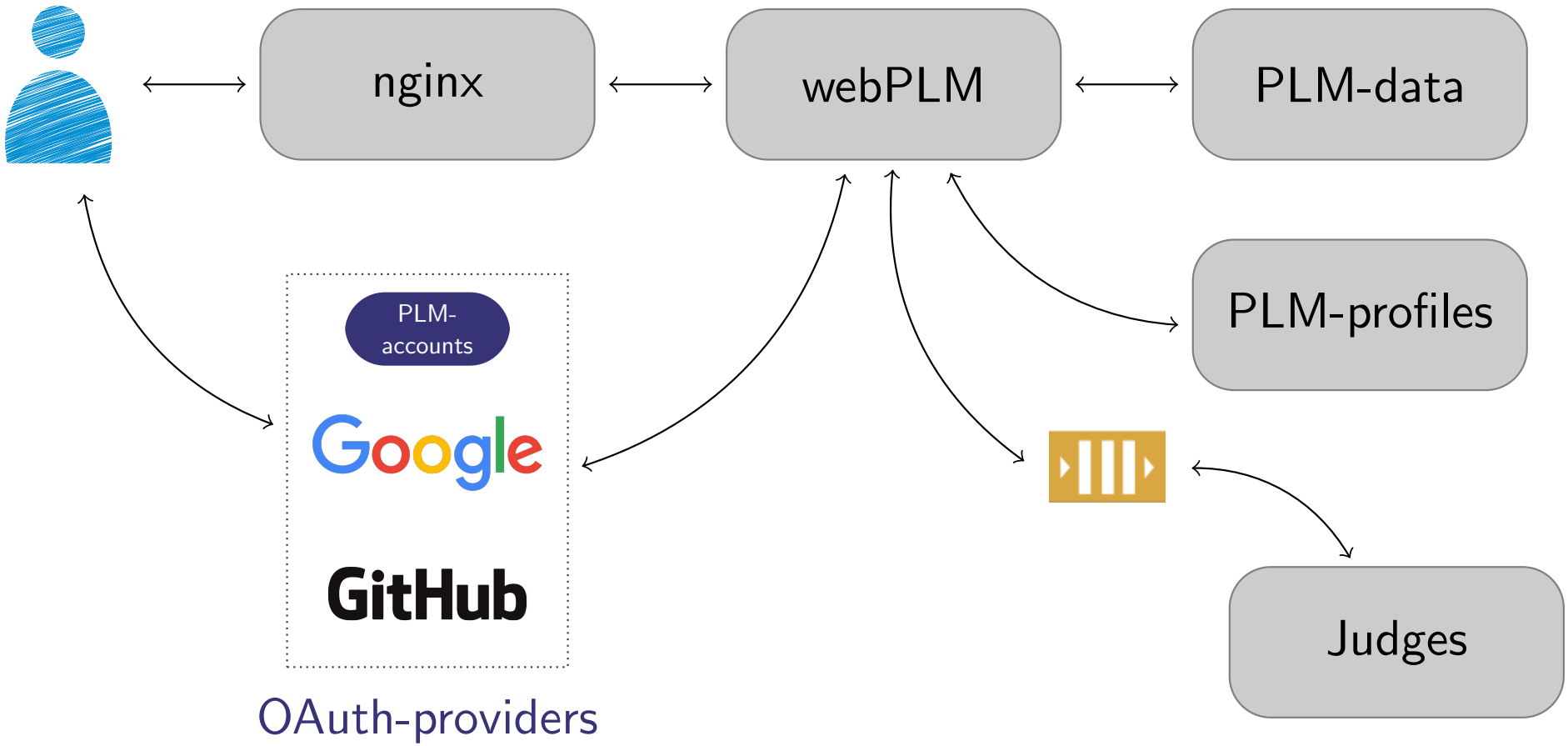
Docker in our case

- Deploy easily all components
- Restart judges automatically
- Limit judges' ressources

Outline

Result

Current architecture



Result

Live-session in TELECOM Nancy

- Rushed to release a stable version before the start of the school year
- Used in TELECOM Nancy in September 2015
- 30 hours of live testing with 100 students

Result

Live-session in TELECOM Nancy

- Rushed to release a stable version before the start of the school year
- Used in TELECOM Nancy in September 2015
- 30 hours of live testing with 100 students

- Engine is (almost) working fine...
- ... but user experience needs to be improved!

Result

Live-session in TELECOM Nancy

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises

Result

Live-session in TELECOM Nancy

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises
- No tools for monitoring set up...

Result

Live-session in TELECOM Nancy

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises
- No tools for monitoring set up...
- ... so the bottleneck is unknown.

Result

Refactor the code

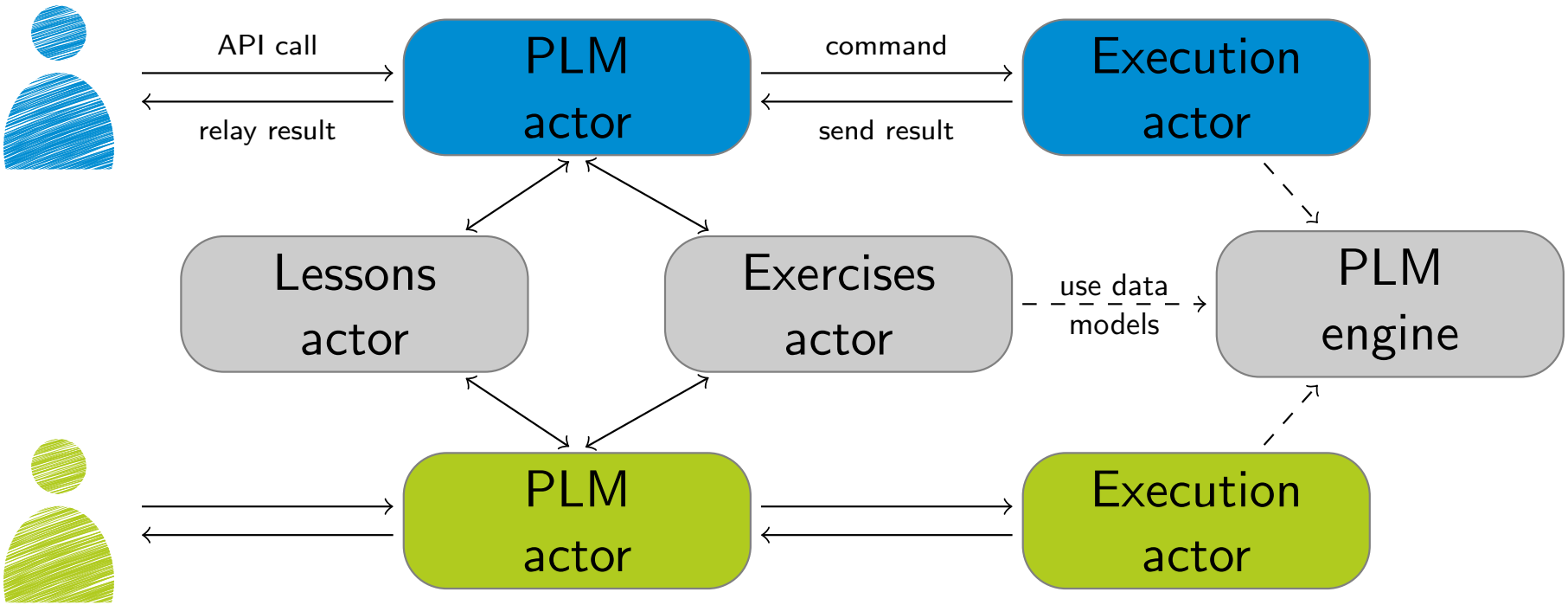
- Needed to clean some parts of the code before further building
- Merged local and centralized mode branches

Outline

Next steps

Extract components from PLM-engine

- Most components are inside PLM-engine
- Updating PLM-engine $\rightarrow$ new version of webPLM and Judge
- Heavy and error prone workflow



- Allow to implement an exercise editor

Next steps

Solve performance issues

- Set up some monitoring tools
- Perform some load testing to identify the bottleneck

Next steps

Sneak peek from the TODO list

- Integrate interns' contributions
- Set up Continuous Deployment
- Support additional programming languages
- Implement a debug mode similar to popular IDEs' ones
- Add features to help teachers to supervise their students
- ...

Questions

Thanks for your attention, any questions?